计算机学院专业必修课

计算机组成

CPU形式建模综合方法概述

高小鹏

北京航空航天大学计算机学院

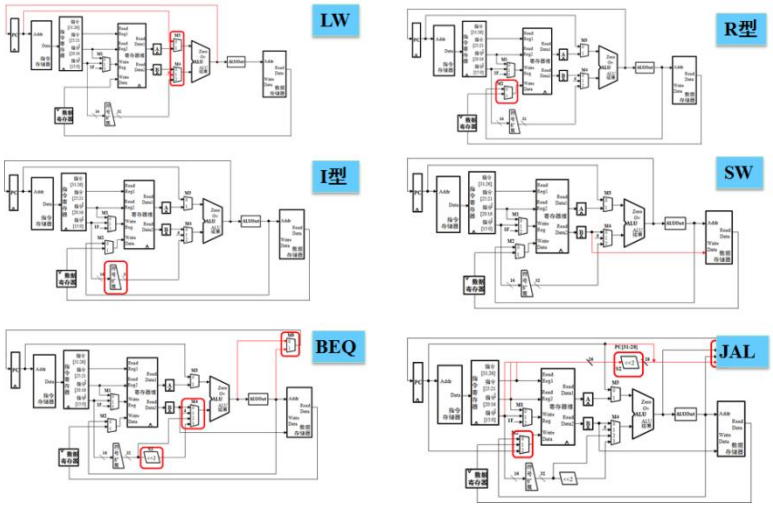
为什么要开发大规模指令集CPU？

- 顶层教学目标：培养本科生的系统能力
  - 本科生开发核心计算系统（CPU、OS、编译器）的能力
- CPU：学习OS和编译器的基础
  - 课程群：计算机组成(2上)、操作系统(2下)、编译技术(3上)
- 技术分析：CPU指令集规模足够大，才能支持OS的运行

10条指令	规模过小，示意型设计
30条指令	初具规模，可运行小型程序 • 指令集规模偏小，难以被成熟工具环境支持
50条指令	可运行简单OS和常规定点程序 • 系统功能指令：中断/异常 • 指令集规模：已可被GCC支持
指令频度分析：①MIPS32指令集；②Linux内核及部分定点程序	

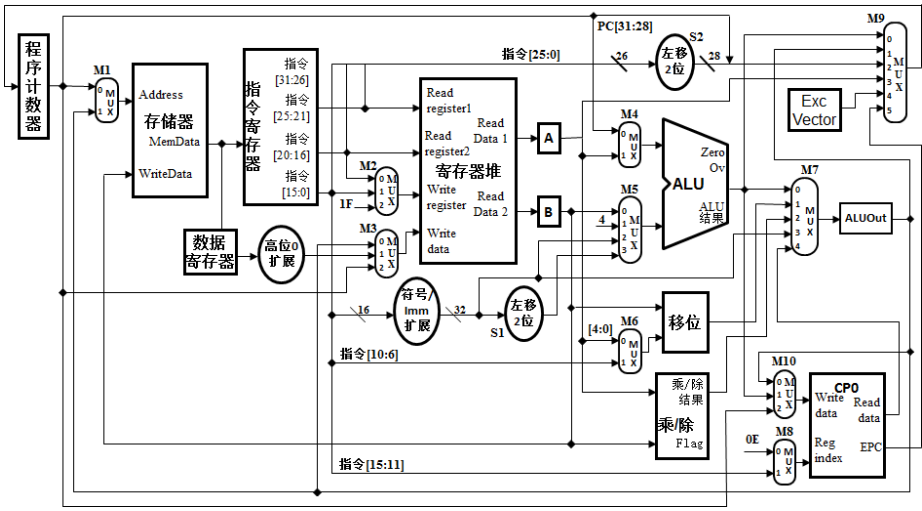
传统的CPU开发教学方法：概述

- 教学挑战：传统教学方法无法满足大规模指令集CPU开发
  - 世界级教材
    - 计算机组成与设计~硬件/软件接口：Patterson；Hennessy
    - 数字设计和计算机体系结构：Harris，Harris
  - 方法基本特点
    - 以功能部件为对象
    - 例举6~8条指令的CPU设计过程，认为可以类推至更大规模指令集



增量式整体性构造CPU

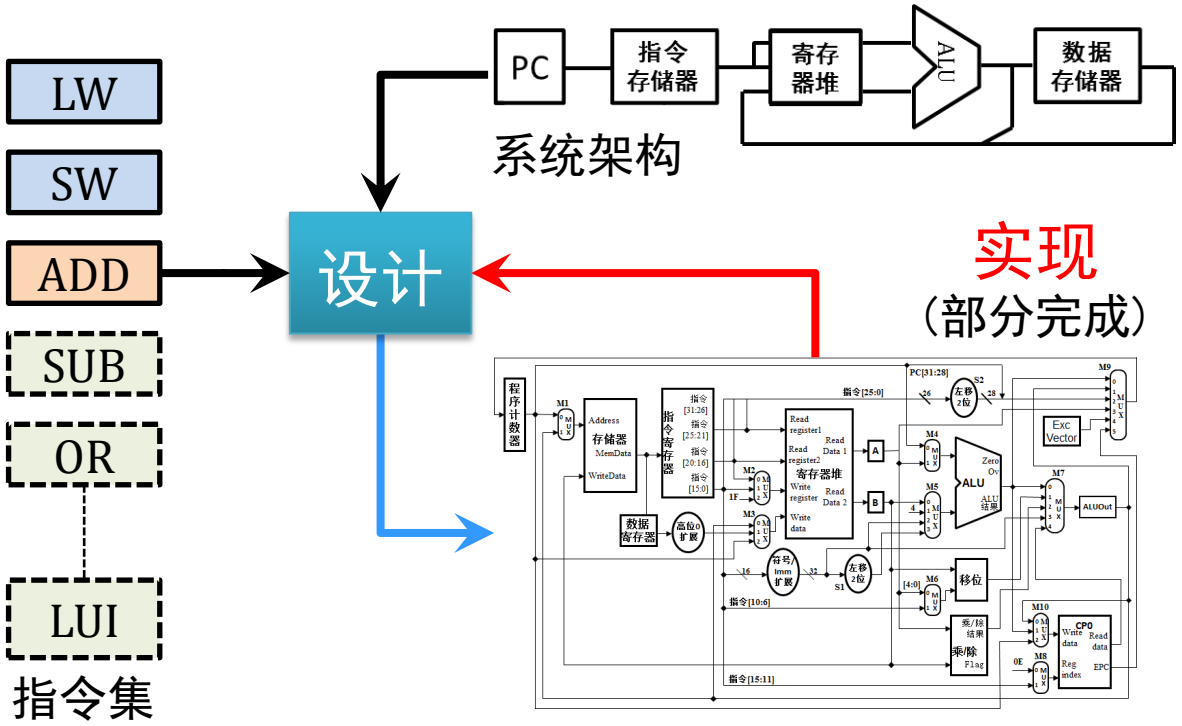
类推



50条指令数据通路

传统的CPU开发教学方法：弊端

- 过程：以系统架构为模型，反复在设计-实现之间进行迭代
- 设计：模型、过程仅在脑中，非显式表达，难以计算
- 实现：每次增量均会导致实现被改变



传统的CPU开发教学方法：弊端

- 教学挑战：传统教学方法无法满足大规模指令集CPU开发
  - 指令构造与整体综合不分离：新增指令对已有设计影响巨大
    - 新增指令导致对拓扑关系的大量修改
    - 一条指令的错误会长期存在并产生影响
  - 设计与实现不分层：设计与实现通常混在编码阶段
    - 不同层次的问题相互干扰，不利于问题剥离
    - 错误传导路径长，不利于问题早期发现
  - 缺乏流水线冲突的完备性分析：无法保障流水线设计的正确性
    - 仅通过例举说明指令间的冲突
    - 缺乏冲突建模，无法100%覆盖所有可能的指令冲突序列

教学实践经验：难以类推至大规模指令集CPU开发

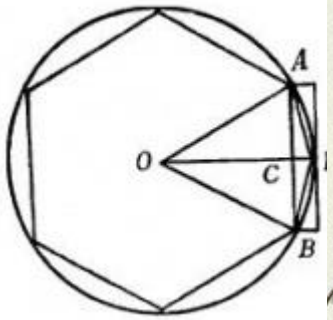
- 单周期、多周期：勉强应对
- 流水线：完全无法确保100%正确

π的启示：图直观性与表达式精确性

- 刘徽（约公元225年—295年）提出了-“割圆术”，计算到圆内接96边形，求得π=3.14。
- 祖冲之（ 公元429年—公元500年）求出π在3.1415926与3.1415927之间。计算到圆内接16384边形。
- 莱布尼茨(1646—1716)提出π的表达式

$$\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}+\cdots\cdots$$

- 当取10000000项，π/4=0.7853981634的前八位保持一致。



信念

- 用形式逻辑的方法可以容易看出，存在某种[指令集]在理论上足以控制和执行任意顺序的操作……从当前的观点出发，选择一个[指令集]时考虑的更多更实际的问题是：[指令集]要求的设备简单性，在实际重要的问题中有明确应用和解决该类问题的速度。
  - Burks,Goldstine &von Neumann, 1947

方法论

- 系统论观点
  - 贝塔朗菲提出“一般系统论”系统作为研究对象，以及功能与结构关系。
  - 系统方法
    - 分析方法：给出系统输入和结构，求取系统输出
    - 综合方法：给出系统功能，构建系统结构。

- 结构主义观点
  - 皮亚杰提出结构主义
  - 系统={子系统1，…，子系统N，子系统间关系，行为}
  - 系统的2种描述方法：结构描述、行为描述

CPU开发工程化方法：思考

- 基本思路：
  - 指令构造与综合过程分离
  - 通过形式建模的方法，实现CPU的多阶段建模及其转换

- 基本要点：
  - 定义了CPU建模语言：确保了对象描述的一致性、可理解性
    - 规格化描述CPU建模过程
  - 定义了建模的若干阶段及其转换方法：基于规则的转换
    - 从指令操作语义、执行过程、数据通路、综合直至代码生成
  - 定义了面向数据流连接的可综合体系架构：综合过程的复杂度是O(n)
    - 显式表达数据通路及连接关系
    - 指令可表达为基于数据通路的多项式，从而综合过程等价为多项式加法

CPU开发工程化方法：过程概述

- 基本过程：形式建模，独立构造，简单综合，快速转换

操作语义

①形式建模

设计输入

Cycle	Stage	Operation	Component	Signal
Cycle1	Fetch (fetch instruction)	IR←IM[PC] PC←PC+4	IR	IRWr: 1
			ALU	ALUOp: ADD
			PC	PCWr: 1
Cycle2	RF (Read operator)	A←RF[rs] ***EXT(Imm16)***	EXT	EXTOp: SE
Cycle3	MA (calculate DM address)	ALUOut←A+EXT	ALU	ALUOp: ADD
Cycle4	MR (read DM)	DR←DM[ALUOut]		
Cycle5	MemWB (write DR to RF)	RF[rt]←DR	RF	RFWr: 1

②独立构造

③简单综合

设计输出

HDL文件

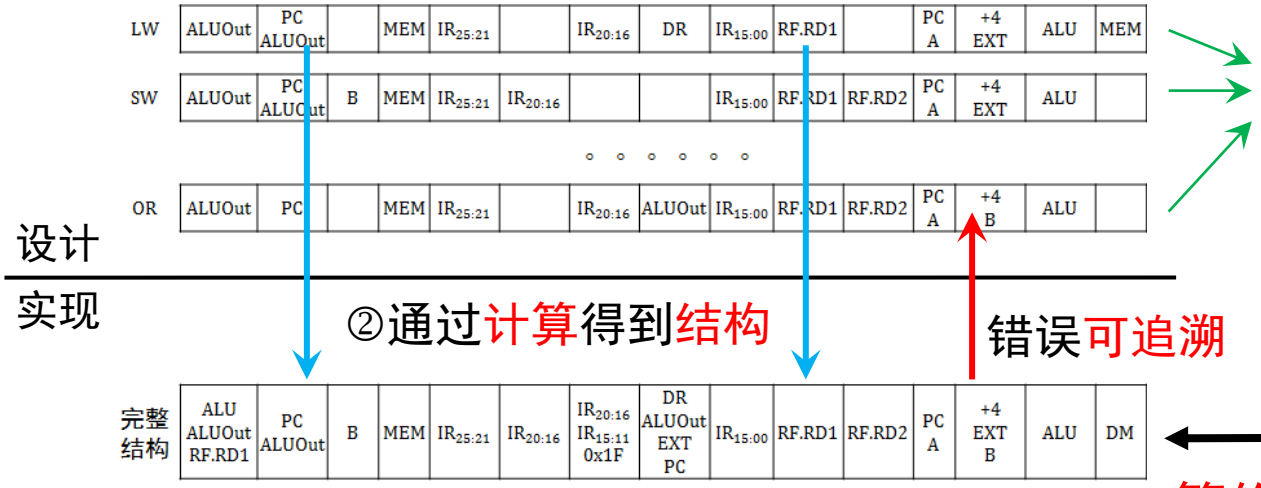
指令 \ 部件	PC	NPC			IM	RF				EXT	ALU		DM
	DI	PC	Imm	RS	A	A1	A2	A3	WD		A	B	A
add	NPC.NPC	PC.DO			PC.DO	IM.D[25:21]	IM.D[20:16]	IM.D[15:11]	ALU.C		RF.RD1	RF.RD2	
sub	NPC.NPC	PC.DO			PC.DO	IM.D[25:21]	IM.D[20:16]	IM.D[15:11]	ALU.C		RF.RD1	RF.RD2	
ori	NPC.NPC	PC.DO			PC.DO	IM.D[25:21]		IM.D[20:16]	ALU.C	IM.D[15:0]	RF.RD1	EXT.Ext	
lw	NPC.NPC	PC.DO			PC.DO	IM.D[25:21]		IM.D[20:16]	DM.RD	IM.D[15:0]	RF.RD1	EXT.Ext	ALU.C
sw	NPC.NPC	PC.DO			PC.DO	IM.D[25:21]		IM.D[20:16]		IM.D[15:0]	RF.RD1	EXT.Ext	ALU.C
slt	NPC.NPC	PC.DO			PC.DO	IM.D[25:21]	IM.D[20:16]	IM.D[15:11]	ALU.C		RF.RD1	RF.RD2	
beq	NPC.NPC	PC.DO	IM.D[15:0]		PC.DO	IM.D[25:21]	IM.D[20:16]				RF.RD1	RF.RD2	
jal	NPC.NPC	PC.DO	IM.D[25:0]	RF.RD1	PC.DO			0x1F	NPC.PC4				
jr	NPC.NPC	PC.DO			PC.DO	IM.D[25:21]							

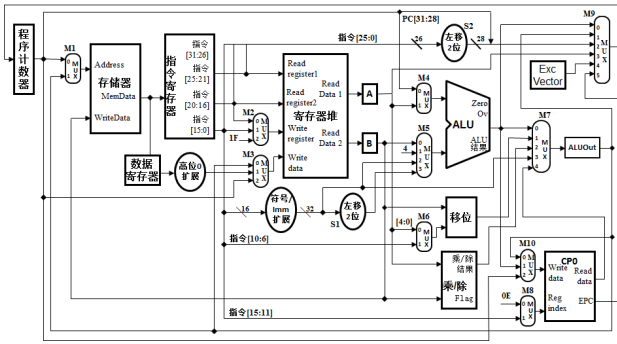
输入信号 \ 部件	PC	NPC			IM	RF				EXT	ALU		DM
	DI	PC	Imm	RS	A	A1	A2	A3	WD		A	B	A
	NPC.NPC	PC.DO	IM.D[25:0]	RF.RD1	PC.DO	IM.D[25:21]	IM.D[20:16]	IM.D[15:11] IM.D[20:16] 0x1F	ALU.C DM.RD NPC.PC4	IM.D[15:0]	RF.RD1	RF.RD2 EXT.Ext	ALU.C

④快速转换

CPU开发工程化方法：优势

- 工程化方法：CPU开发复杂度低





优势：

1.设计：与实现解耦

2.综合：指令集整体综合，可计算

3.排错：错误可追溯